# Aula 1 - Introdução ao Qt

**Resumo:**

- Instalação
- Projetos com o `QtCreator` e `QtDesigner`
- Exemplo: `notepad`

## 1.1. Instalação

`Qt` (pronunciado cute) é uma aplicação multiplataforma para desenvolvimento de Interfaces Gráficas baseadas em C++ ou QML (uma linguagem semelhante ao CSS/javascript). O QtCreator é o ambiente de desenvolvimento suportado pela própria aplicação (sendo possível usar outros ambientes de desenvolvimento). Desenvolvido originalmente pela divisão Nokia Qt, da empresa norueguesa Trolltech, é atualmente uma iniciativa de software livre mantido pelo QtProject e oferecendo várias alternativas de licenciamento livre ou comercial.

O Qt pode ser descarregado no endereço seguinte (http://qt-project.org/downloads) para várias plataformas (Windows, Linux, MacOs, Android, iOs, …). De notar que a instalação ocupa bastante espaço em disco (da ordem de 3GB no Windows por exemplo).

De notar que este tutorial é adaptado de um tutorial Qt adaptado para utilização com o QtCreator.

Qt Notepad example: http://qt-project.org/doc/qt-4.8/gettingstartedqt.html
Notepad with QtWidgets: http://qt-project.org/doc/qt-5/gettingstartedqt.html

## 1.2. Criação de um projeto no Qt Creator

Abra o QtCreator e crie uma nova aplicação com do tipo QT Widgets Application com o nome notepad. A classe base do projeto deve ter o nome notepad e ser do tipo `QMainWindow`. Note que o Qt Creator cria automaticamente os ficheiros seguintes (eventualmente com nomes diferentes):

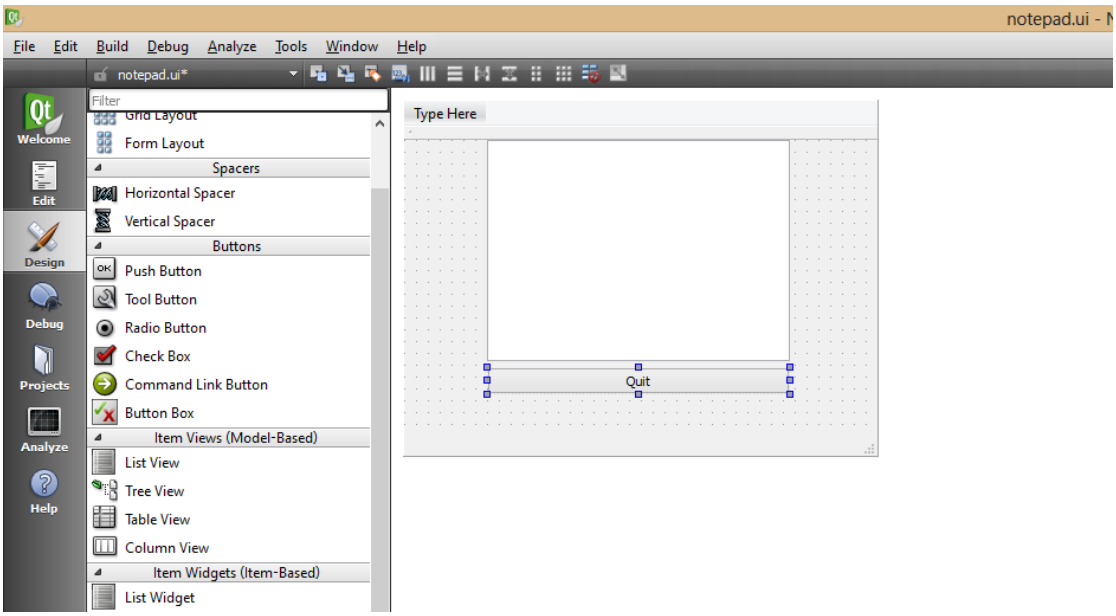
- `notepad.pro` – O ficheiro do projeto.
- `main.cpp` – ficheiro fonte principal da aplicação.
- `notepad.cpp` – ficheiro fonte da classe notepad do widget notepad.
- `notepad.h` – cabeçalho da classe notepad do widget notepad.
- `notepad.ui` – a janela de interface widget notepad.

Execute o programa e veja o que ocorre.

### 1.1. Desenho da interface

Abra o ficheiro `notepad.ui` que define a interface da aplicação em XML. Ao abrir o ficheiro no QtCreator é automaticamente usado o QtDesigner (aplicação gráfica de desenho de interface do Qt). Ao compilar a aplicação o Qt cria automaticamente o ficheiro `ui_notepad.h` com o código C++ correspondente à interface criada no QtDesigner.

No QtDesigner adicione na interface uma caixa de texto (`QTextEdit`) e um botão (`QPushButton`). Mude o texto no botão para `Quit` e o nome para `quitButton`. Selecione os dois elementos e use a opção `Layout Vertically` para os alinhar obtendo uma interface semelhante a figura seguinte:




Utilize um outro programa (wordpad por exemplo) para observar o ficheiro XML .ui que descreve a interface. Pode também observar os ficheiros `notepad.h` e `notepad.cpp` onde será programada a lógica associada a interface gráfica.

### 1.2. Interação com o botão

O Qt utiliza o mecanismo "`signals and slots`". Os sinais são emitidos pelos controlos quando um dado evento ocorrer e os `slots` são funções que são chamadas em resposta ao evento (não passa de outros nomes para o esquema típico usado em desenho de interface de "`events and callbacks`").

Para adicionar um evento e uma função ao botão, no QtDesigner deve clicar no botão com a tecla direita e no menu contextual selecionar `Go to slot ->clicked()`. Isso vai gerar uma função nos ficheiros `notepad.h` e `notepad.cpp`. Na função criada no ficheiro notepad.cpp, adicione o código `qApp->quit()` para sair da aplicação.

Corra a aplicação e verifique que o botão permite fechar a aplicação.

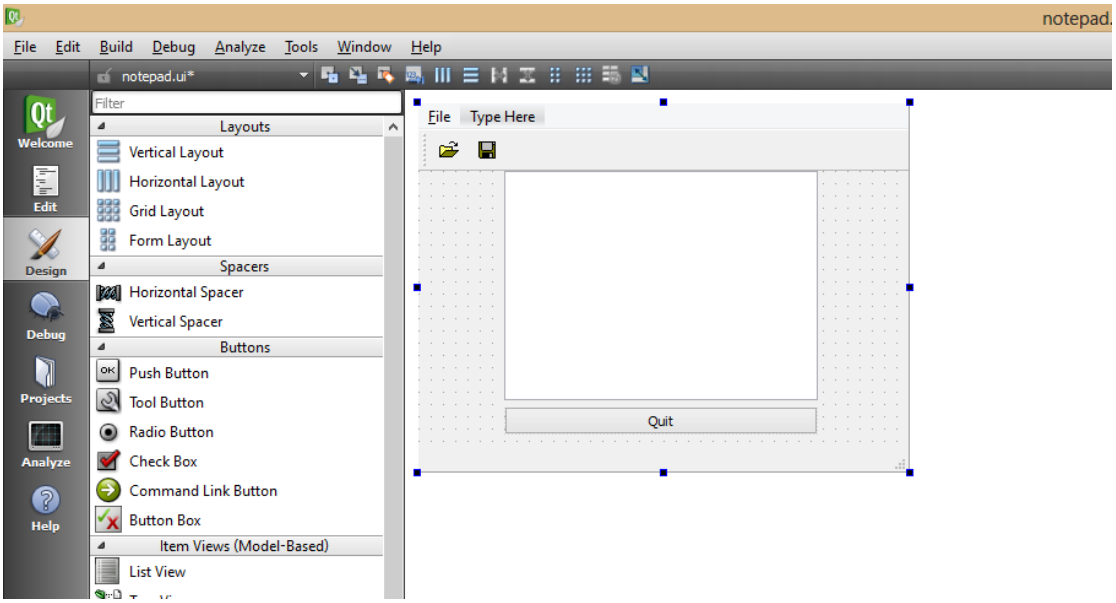
### 1.3. Adição de um menu

É muitas vezes útil que o mesmo `slot` seja invocado por vários elementos (por exemplo um menu e um `icon` que invocam a mesma função). Para permitir esse

comportamento, o Qt disponibiliza as chamadas `QAction` que podem ser associados a vários controlos (ser associados ao mesmo `slot`). Por exemplo as classes `QMenu` e `QToolbar` podem criar itens de um menu e botões de uma `toolbar` para a mesma ação. Como exemplo, vamos adicionar um menu na interface e `icons` para as mesmas ações na `toolbar`.

No QtDesigner, acrescente no menu os elementos `Files->Open` e `File->Save` (Faça double-click no placeholder "Type Here"). Associe a essas ações os `icons save` e `open` fornecidos (opção `choose file` depois de clicar nos … juntos ao `icons` nas propriedades). Associe ainda as teclas de atalho adequadas (caracter & antes da letra a usar).

Finalmente coloque essas duas opções na `QMainToolbar`. Para tal, deve arrastar as ações a partir do `Action Editor` obtendo algo semelhante a figura seguinte.




## 1.4. Abertura de um ficheiro

Vamos utilizar uma `QFileDialog` para pedir ao utilizar qual o ficheiro ao abrir ao selecionar a opção `open`. Para tal adicione o `signal "triggered"` a opção `Open` e coloque o código seguinte na função:

```
      void Notepad::on_actionOpen_triggered()
      {
          QString fileName = QFileDialog::getOpenFileName(this,
tr("Open File"), QString(),
                  tr("Text Files (*.txt);;C++ Files (*.cpp *.h)"));

          if (!fileName.isEmpty()) {
              QFile file(fileName);
              if (!file.open(QIODevice::ReadOnly)) {
                  QMessageBox::critical(this, tr("Error"), tr("Could
not open file"));
                  return;
              }
              QTextStream in(&file);
              ui->textEdit->setText(in.readAll());
              file.close();
          }
```

```
}
```

Este código permite abrir um ficheiro (filtrando pelas extensões do tipo .txt, .cpp or .h) e ler o seu conteúdo para a `QTextEdit` da interface.

Não se esqueça de adicionar os incudes seguintes antes de compilar

```
#include <QFileDialog>
#include <QFile>
#include <QMessageBox>
#include <QTextStream>
```

### 1.5. Salvar um ficheiro

Da mesma a forma utilize o código seguinte para permitir salvar o conteúdo da `QTextEdit` num ficheiro de texto.

```
void Notepad::on_actionSave_triggered()
{
    QString fileName = QFileDialog::getSaveFileName(this,
tr("Save File"), QString(),
            tr("Text Files (*.txt);;C++ Files (*.cpp *.h)"));

    if (!fileName.isEmpty()) {
        QFile file(fileName);
        if (!file.open(QIODevice::WriteOnly)) {
            QMessageBox::critical(this, tr("Error"), tr("Could
not save file"));

        } else {
            QTextStream stream(&file);
            stream << ui->textEdit->toPlainText();
            stream.flush();
            file.close();
        }
    }
}
```

### 1.6. Validação antes de sair

Modifique ainda o programa para pedir confirmação antes de sair do programa (clique no botão) permitindo validar a escolha através de uma `QMessageBox`.

```
void Notepad::quit()
{
      QMessageBox messageBox;
      messageBox.setWindowTitle(tr("Notepad"));
      messageBox.setText(tr("Do you really want to quit?"));
      messageBox.setStandardButtons(QMessageBox::Yes |
QMessageBox::No);
      messageBox.setDefaultButton(QMessageBox::No);
      if (messageBox.exec() == QMessageBox::Yes)
            qApp->quit();
}
```

### 1.7. Mudança da fonte de texto

Apesar de muito útil o QtDesigner tem várias limitações e não permite fazer todo o tipo de manipulação. Por exemplo é impossível adicionar através do QtDesigner controlos mais avançados as `mainToolbar`.

Como exemplo, adicione uma `QFontComboBox` a `mainToolbar`. Isso só pode ser feito manipulando diretamente o código. Para tal deve:

- Definir uma variável do tipo `QFontComboBox` como atributo da classe notepad.cpp
- No construtor da classe inicializar essa variável, adiciona-la a `toolbar` e ainda associa-la a um `slot`, usando por exemplo, o código seguinte:

```
myCombo = new QFontComboBox();
ui->mainToolBar->addWidget(myCombo);
connect(myCombo,SIGNAL(currentFontChanged(QFont)),this,SLOT(fon
tChanged()));
```

- Definir a função associada a esse `signal` que permite neste caso atualizar a fonte do texto:

```
void Notepad::fontChanged()
{
     ui->textEdit->setCurrentFont(myCombo->currentFont());
}
```

- Adicionar o protótipo da função na classe `notepad.h`.

### 1.8. Outras opções de texto

Faça as modificações necessárias para permitir mudar o tamanho do texto na caixa de texto (escolhendo por exemplo entre 3 tamanhos diferente: 10, 12 e 14pt). Adiciona ainda a possibilidade de especificar o negrito, itálico e sublinhado.